

SOLUTIONS TO THE PROBLEM

A deadlock avoiding system of the present invention comprises:
periodic monitoring means for periodically determining a deadlock
with serially reusable resources; a task detection means for
5 detecting a deadlocked task; batch releasing means for comparing
execution levels of the detected deadlocked tasks, for releasing
all serially reusable resources locked by tasks having low
execution levels, and for preferentially running tasks having high
execution levels; context saving means for saving contexts when
10 the serially reusable resources locked by tasks having low
execution levels are released; context restoring means for
restoring the contexts when the tasks having low execution levels
are re-started; and batch securing means for collectively
re-securing the released serially reusable resources.

⑩ 日本国特許庁(JP)

⑪ 特許出願公開

⑫ 公開特許公報(A) 平4-85632

⑬ Int. Cl.⁵

G 06 F 9/46

11/30

12/00

15/16

識別記号

3 4 0 G

3 1 3 Z

3 0 5 G

5 3 5 A

4 7 0 A

庁内整理番号

8120-5B

8120-5B

7165-5B

8944-5B

9190-5L

⑭ 公開 平成4年(1992)3月18日

審査請求 未請求 請求項の数 1 (全5頁)

⑮ 発明の名称 デッドロック回避方式

⑯ 特 願 平2-199008

⑰ 出 願 平2(1990)7月30日

⑱ 発 明 者 大 野 重 幸 東京都港区芝5丁目7番1号 日本電気株式会社内

⑲ 出 願 人 日本電気株式会社 東京都港区芝5丁目7番1号

⑳ 代 理 人 弁理士 山下 穰平

明 細 書

1. 発明の名称

デッドロック回避方式

2. 特許請求の範囲

定期的に逐次使用可能資源のデッドロックを判定する定期監視手段と、デッドロック中のタスクを検出するタスク検出手段と、検出されたデッドロック中のタスクの実行レベルを比較し、実行レベルの低いタスクがロックしているすべての逐次使用可能資源を解放し、実行レベルの高いタスクを優先して走行させるための一括解放手段と、実行レベルの低いタスクの逐次使用可能資源解放時のコンテキストを退避するためのコンテキスト退避手段と、実行レベルの低いタスクの再スタート時にコンテキストを回復するためのコンテキスト回復手段と、解放した逐次使用可能資源を一括して再確保するための一括確保手段とを備えていることを特徴とするデッドロック回避方式。

3. 発明の詳細な説明

(産業上の利用分野)

本発明は、密結合構成のマルチプロセッシングシステムにおける逐次使用可能資源のデッドロック回避方式に関する。

(従来の技術)

従来、密結合マルチプロセッシングシステムにおいて、同時使用不可であるが逐次使用の可能な資源、すなわちSRR (Serially Reusable Resource) のデッドロックが発生した場合には、デッドロックを起こしている双方のタスクの実行レベルを比較して実行レベルが低い方のタスクが確保しているSRRをすべて解放し、実行レベルが低い方のタスクを最初のSRR確保以前の状態に戻すことによってデッドロックを回避していた。

第4図は上述した従来の方式の具体例を示す図であり、CPU50のタスクAがSRR52を使用するため他タスクからの使用を禁止(ロック)し、またCPU51のタスクBがSRR53をロックしているときに、タスクAがSRR53を、タスクBがSRR52をロックしようとした場合に、双方の使用禁止(ロック)を解除できなくな

り、デッドロックとなる。

ここで、タスクAよりタスクBの実行レベルの方が低い場合、タスクBがロックしているSRR 53を解放し、タスクBをSRR 53のロック以前の状態に戻すことによって、デッドロックの回避を行っていた。

〔発明が解決しようとする課題〕

しかしながら、上述した従来の方式では、SRRのデッドロックが発生すると実行レベルが低いタスクは最初にSRRを確保した時点まで戻されてしまうことになり、SRRのデッドロックが発生した時点までに行った処理が無駄になってしまうという欠点があった。

本発明は、上記のような従来の欠点を改善したもので、その目的は、SRRのデッドロックを起こしている双方のタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしている全てのSRRを解放し、実行レベルの低いタスクを再スタートするときこのタスクがデッドロック発生時までに行った処理が無駄になるのを防止す

ることの可能なデッドロック回避方式を提供することにある。

〔課題を解決するための手段〕

本発明のデッドロック回避方式は、定期的に逐次使用可能資源のデッドロックを判定する定期監視手段と、デッドロック中のタスクを検出するタスク検出手段と、検出されたデッドロック中のタスクの実行レベルを比較し、実行レベルの低いタスクがロックしているすべての逐次使用可能資源を解放し、実行レベルの高いタスクを優先して走行させるための一括解放手段と、実行レベルの低いタスクの逐次使用可能資源解放時のコンテキストを退避するためのコンテキスト退避手段と、実行レベルの低いタスクの再スタート時にコンテキストを回復するためのコンテキスト回復手段と、解放した逐次使用可能資源を一括して再確保するための一括確保手段とを有している

〔作 用〕

逐次使用可能資源(SRR)のデッドロックが発生した場合に、実行レベルの低いタスクを一時

的にストップさせて、これがロックしているSRRをその間解放し、再スタート時には、一括してSRRを再確保することで、実行レベルの低いタスクがデッドロック発生時までに行った処理が無駄になるのを防止している

〔実施例〕

次に本発明について図面を参照して説明する。

第1図は本発明の一実施例のブロック図である。本実施例は、一定時間毎に起動しデッドロックを判定する定期監視手段11と、デッドロックと判定された場合、デッドロックを起こしているタスクを検出するタスク検出手段12と、検出された双方のタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしているすべてのSRRを一括解放し、実行レベルの高いタスクを優先して走行させるためのSRR一括解放手段13と、実行レベルの低い方のタスクのSRR解放時のコンテキスト(タスクが走行するための環境)を退避するためのコンテキスト退避手段14、実行レベルの低い方のタスクが走行可能になった際、タ

スクを再スタートするために、退避してあったコンテキストを回復するためのコンテキスト回復手段15と、SRR一括解放手段13で解放したすべてのSRRを一括して再確保するためのSRR一括確保手段16とで構成されている。

次にこのような構成におけるSRRのデッドロック回避処理動作について説明する。

SRRのデッドロック回避処理を行うために、第2図に示すように、SRR1つづつに対応する情報域INFを設ける。情報域INFには、デッドロックを判定するためのカウンタと、デッドロックしているタスクを検出するための2つのタスクID格納域とを設ける。

SRRをロックした時にロックしたSRRのSRRID(SRR識別子)をタスクコントロールブロックTCBからポイントされるSRRロックリストに設定する。さらに、そのSRRに対応する情報域INFの1つのタスクID格納域にロックしたタスク、例えばAのタスクID(タスク識別子)を設定する。既にロックしてあるSRR

に、他のタスク、例えばBがロックしようとした場合、無限リトライの場合に限り、ある回数以上リトライすれば、そのSRRに対応する情報域INFのもう1つのタスクID格納域にそのタスクBのタスクIDを設定する。

またSRRをアンロックした時には、そのSRRに対応する情報域INFをクリアしておく。

定期監視手段11は定期的に起動されるタスクであり、起動されると、先ず、ロックされているSRRがあるか判定（情報域INFがクリアされていればロックされていない）し、無ければ再びウエイトする。ロックされているSRRがあれば、そのSRRに対応する情報域INFのカウントを1アップし、それをデッドロック判定しきい値と比較し、デッドロック判定しきい値より大きければ、そのSRRをデッドロックとみなし、タスク検出手段12を呼び出す。デッドロック判定しきい値以下なら再びウエイトする。

タスク検出手段12は、定期監視手段11でデッドロックとみなされたSRRに対応する情報域

INFの、タスクIDからデッドロックになっているタスクを検出する。

SRR一括解放手段13でそのタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしているSRRをSRRロックリストより求め、すべて解放する

このとき、コンテキスト退避手段14によって実行レベルの低いタスクのコンテキストをタスクコントロールブロックTCBからポイントされる領域に退避し、該タスクは、実行待ち状態になる。

その後、実行レベルの高い方のタスクの処理が終了し、実行レベルに低い方のタスクが走行可能になった場合、コンテキスト退避手段14によって退避されているコンテキストをコンテキスト回復手段15によって回復し、さらに、SRR一括解放手段13で解放したすべてのSRRをSRR一括確保手段16によって確保することによって、実行レベルの低いタスクをデッドロック発生時点から再度走行させることができる。

第3図は本実施例による上述した処理方式の具

体例を示す図である。

第3図を参照すると、デッドロックが発生すると、タスクAよりタスクBの実行レベルの方が低い場合には、タスクBがロックしているSRR53を解放し、タスクBのSRR52のロック失敗時のコンテキストを退避する。タスクAの処理が終了し、タスクBが走行可能になった際、SRR53を確保し、退避してあったコンテキストを回復することによって、SRR53のロック時からSRR52のロック時までの間にタスクBが行った処理が無駄になるのを防ぎタスクAを優先的に走行させてデッドロックの回避を行うことができる。

〔発明の効果〕

以上説明したように、本発明は、SRRのデッドロックが発生した場合、実行レベルの低い方のタスクを一時的にストップさせ、そのロックしているSRRをその間解放し、再スタート時に一括してSRRを再確保することによって、実行レベルの低いタスクがデッドロック発生時までに行った処理が無駄になることを防止し、デッドロック

を回避することができるという効果がある。

4. 図面の簡単な説明

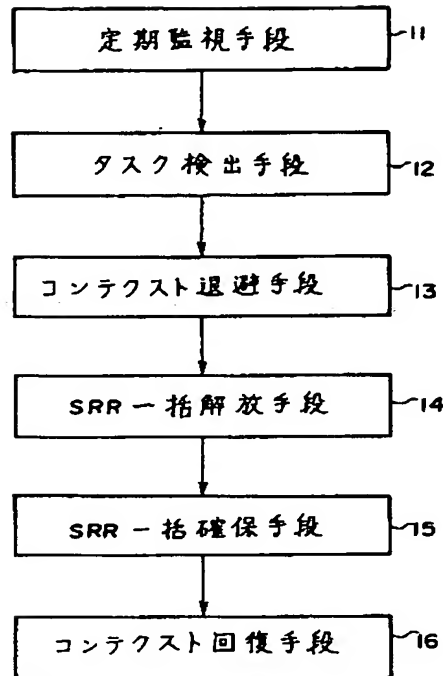
第1図は本発明の一実施例のブロック図、第2図はSRRとタスクとの関連を示す図、第3図は本発明によるデッドロック回避処理の具体例を示す図、第4図は従来のデッドロック回避処理の具体例を示す図である。

第1図において、

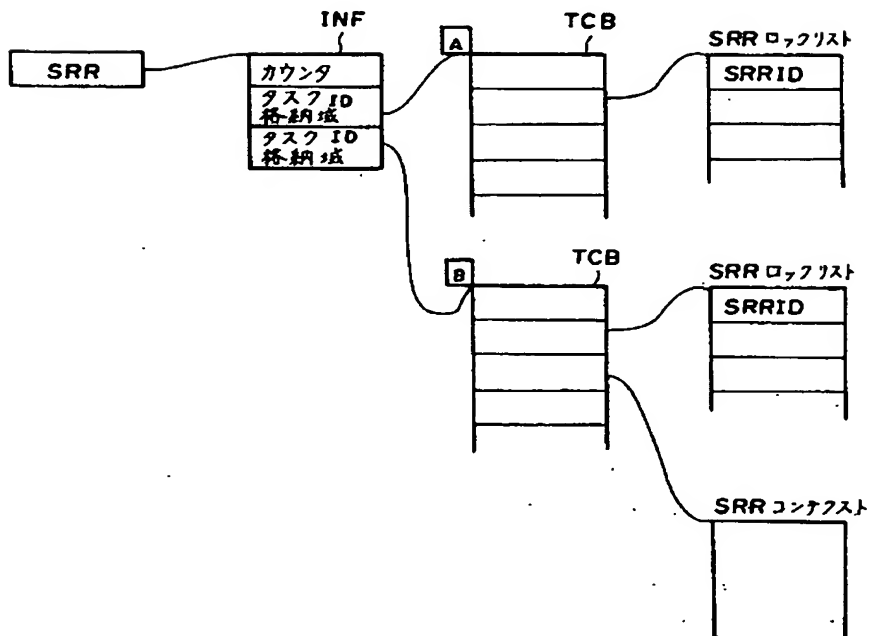
11…定期監視手段、12…タスク検出手段、13…コンテキスト退避手段、14…SRR一括解放手段、15…SRR一括確保手段、16…コンテキスト回復手段。

代理人 弁理士 山下 穰 平

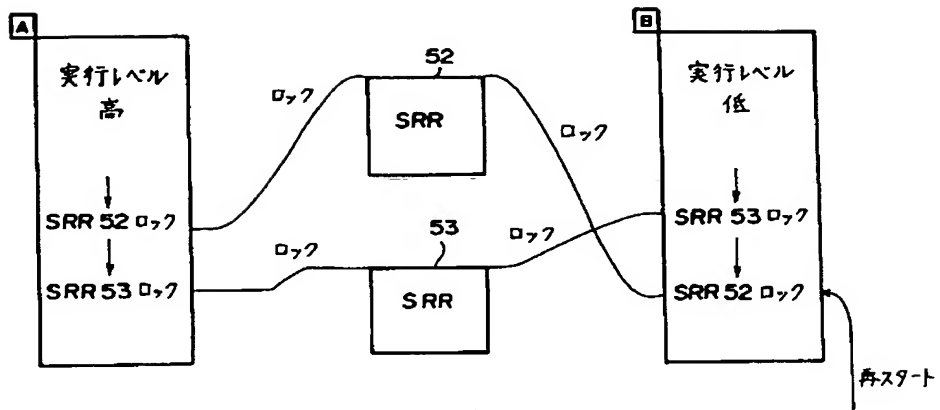
第 1 図



第 2 図



第 3 図



第 4 図

